MARKOV CHAIN ALGORITHMS FOR EMERGENT PHENOMENA IN SELF-ORGANIZING PARTICLE SYSTEMS

A Thesis
Presented to
The Academic Faculty

By

Cem Gokmen

In Partial Fulfillment
of the Requirements for the
Undergraduate Research Option
in the School of Computer Science

Georgia Institute of Technology

December 2018

MARKOV CHAIN ALGORITHMS FOR EMERGENT PHENOMENA IN SELF-ORGANIZING PARTICLE SYSTEMS

Approved by:

Dr. Dana Randall, Advisor
School of Computer Science
Georgia Institute of Technology

Dr. Richard (Yang) Peng
School of Computer Science
Georgia Institute of Technology

Date Approved: December 14, 2018

To live! Like a tree, alone and free, and like a forest, in brotherhood - this longing is ours.

Nazim Hikmet Ran

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

We investigate Markov chain algorithms that can accomplish global emergent phenomena in a distributed setting, namely in *self-organizing particle systems*, an abstraction of programmable matter. These particle systems are composed of individual computational units known as *particles* that each have limited memory, strictly local communication abilities, and modest computational power, and which collectively solve system-wide problems of movement and coordination. We introduce fully distributed, asynchronous, stochastic algorithms for two problems: *separation* and *alignment*, both of which we formally define. We then rigorously analyze the correctness and convergence of our distributed, stochastic algorithms by leveraging techniques from Markov chain analysis. Our theoretical results are complemented by simulations.

CHAPTER 1
INTRODUCTION

Many programmable matter systems have recently been proposed and realized—modular and swarm robotics, synthetic biology, DNA tiling, and smart materials form an incomplete list—and each is often tailored toward a specific task or physical setting. With implementations of the concept of *programmable matter* now available, the focus has shifted to being able to harness the capabilities of these systems. With a large variety of goals for these systems to accomplish, and with very little information for each individual agent to use to this end, the ability to devise the algorithms to produce certain emergent phenomena has become the focus of research.

In this thesis, we investigate algorithms for *self-organizing particle systems*, an abstraction of programmable matter that allows us to go beyond a single implementation, with the aim of introducing and analyzing robust, scalable and practical methods of producing key emergent phenomena (*separation* and *alignment*) on self-organizing particle systems. Based on Markov chains, we show and analyze how our stochastic algorithms, using just local moves, are able to reach different global goals in a robust and efficient manner.

1.1 Self-Organizing Particle Systems

With many different implementations of programmable matter from a large variety of disciplines, in order to be able to discuss general approaches and results, an abstraction is required. In self-organizing particle systems, which is one such abstraction, a set of *particles*, which are computing agents equipped with finite memory and local information, occupy the vertices of some graph, and are given the ability to actively move on edges of the graph to adjacent vertices. The *particles* perform computations and moves asynchronously. Using such an abstraction, it is possible to discuss and rigorously prove algorithms for self-

organizing particle systems to understand the general capabilities of Programmable Matter.

A variety of algorithms were developed for self-organizing particle systems. Examples provide abilities such as leader election between particles [1], shape formation [2], arithmetic computation [3], and object coating [4]. A common property of these is that they are deterministic algorithms, and that they require particles to have special roles as *seeds* or *leaders*. Therefore, they are not robust to failures and modifications in the system (e.g. particles breaking down, being removed, more particles being added, etc.). These algorithms are also increasingly hard to develop and prove the correctness of, due to their intricate nature with usually multiple steps and subroutines. The necessity of waiting for leadership also prevents these algorithms from fully harnessing the distributed nature of the self-organizing particle system concept.

1.2 The Stochastic Approach

The stochastic approach to self-organizing particle systems was first put forward in [5] where the authors present and prove the correctness of a stochastic algorithm for *compression* in a particle system, and was further validated in [6] which features a bridging algorithm of similar nature. The algorithms are based on Markov chains that were designed by the authors to converge to an ideal stationary distribution, which, in the case of compression, is a configuration with the lowest possible perimeter. This approach is motivated by studies from statistical physics that investigate the local micro-behavior causes of global macroscopic phenomena [7, 8, 9]. Since the underlying Markov chain only allows transitions by local moves (i.e. particles can only move to immediately adjacent locations in one step of the Markov chain) with local knowledge (i.e. only information from the immediate neighborhood of the particle is used), it can be converted to a distributed algorithm to be run asynchronously on the particle system. Moreover, since the algorithm is based directly on a Markov chain, existing methods of analysis for Markov chains can be used to make guarantees about the correctness of the algorithm. Stochastic algorithms also have

the benefit of making the system fully stateless where each particle has the same role, thus making it directly scalable, and the algorithms less complicated. Other benefits include trivial robustness against failures of a subset of particles as well as against unexpected additions to the particle system: since the algorithms are stateless, they will perform the same regardless of particles being added to or removed from the system.

CHAPTER 2

BACKGROUND

2.1 The Geometric Amoebot Model

Even though the model of a *self-organizing particle system* abstracts away implementation details of the Programmable Matter systems, its lack of sufficiently detailed definitions for particle capabilities and the underlying graph creates a need for a *model* with precisely defined rules for use in formal proofs. One such model is called the *Geometric Amoebot Model*, introduced in [10]. This model is used in introductions of algorithms including object coating [11], leader election [1], shape formation [2], and others.

In this model, particles are placed on vertices of a infinite triangular lattice graph, G_Δ. Each particle is capable of moving into adjacent positions on the lattice through expansion and contraction: a particle expands into an adjacent position, occupying both its original position and this new position as well as the edge between the two positions, then contracts into either position. While particles are anonymous, they can uniquely identify neighbor particles, they can communicate with neighboring particles, and they share a notion of *chirality*, which allows all particles to count their neighbors in a common clockwise or counter-clockwise order, despite not having a notion of global orientation.

Each particle has a constant-sized memory which can be written to and read from both

(a) (b)

Figure 2.1: (a) A section of the triangular lattice G_Δ; (b) expanded particles (each denoted by its two occupied adjacent locations in G_Δ and a thick line in between) and contracted particles (occupying one location).

by the particle itself and its neighbors. Due to the constant-sized memory limitation, the particle system cannot keep track of more than a constant number of other particles, or maintain arbitrarily large numbers, as a result becoming incapable of storing any information about the size or shape of the overall system. Particles are able to perform *atomic* actions, which are limited to some arbitrary amount of computation about the local environment, plus an expansion or contraction move. These atomic actions occur asynchronously, in a distributed manner, and all conflicts resulting from expansion into the same location by multiple particles are resolved in an arbitrary manner.

In our model, we assume that the system progresses through *atomic actions*, in line with the standard asynchronous model of computation from distributed computing (see, e.g., [12]). Previous results from the model state that there exists a sequential, synchronous ordering of atomic actions that can produce the same end result as any asynchronous execution of atomic actions. In our model, we consider one particle activation consisting of some bounded amount of computation and at most one expansion or contraction move to be comprise an atomic move. Assuming that concurrent memory writes and simultaneous moves into the same lattice point are arbitrarily resolved, it thus suffices to consider particle activations as a sequence when analyzing our algorithm.

2.2 Terminology for the Amoebot Model

Now, we introduce notation and terminology that will be used throughout this thesis.

We define an *arrangement* as the set of tuples that contain the location and flags of each particle. For example, in the Separation model, each particle has a position from the set Γ, and while each particle is anonymous, it carries identifiable information in the form of a flag $c \in C = \{0, 1\}$ representing its color. In that case, an arrangement is in the form of $\{(\Gamma \times C)\}^*$. As a result, two arrangements are the same if two particles of the same color have swapped locations between the two, but not the same if two particles of different colors have swapped locations.

We define a *configuration* to be an equivalence class of arrangements, where we consider two arrangements equivalent if they are translations of one another. If configuration σ is a rotation of configuration τ, we still consider σ and τ to be distinct configurations.

Capital letters will refer to particles and lowercase letters refer to locations on the lattice, e.g., "particle P at location ℓ." For a particle P and location ℓ, we use $N(P)$ and $N(\ell)$ to denote the set of particles that neighbor P or ℓ, that is, particles whose locations share a lattice edge with P or ℓ, respectively. Similarly we use $n(P)$ and $n(\ell)$ to denote the set of neighboring locations of a particle or location, respectively. When discussing the union of the neighborhoods of two locations or particles such as $N(\ell \cup \ell')$, we mean $(N(\ell) \cup N(\ell')) \setminus \{\ell, \ell'\}$.

An *edge* of a configuration is an edge of the underlying lattice whose both incident vertices are occupied by particles. $e(\sigma)$ indicates the number of edges of a configuration σ. We say that a configuration is connected if all pairs of particles are connected, i.e. a path can be found between them. When we discuss cycles on a configuration, we mean a cycle on the lattice whose vertices are occupied by particles and whose interior has at least one unoccupied location. Such components of unoccupied adjacent locations encircled by cycles are called *holes*.

We specifically consider connected configurations with no holes, and our algorithm, if starting at such a configuration, will maintain these properties.

2.3 Markov Chains

The distributed protocol for particle compression we present is based on a Markov chain, that is, a memoryless stochastic process defined on a finite set of states Ω. The transition matrix P on $\Omega \times \Omega \to [0, 1]$ is defined so that $P(x, y)$ is the probability of moving from state x to state y in one step, for any pair $x, y \in \Omega$. The t-step transition probability $P^t(x, y)$ is the probability of moving from x to y in exactly t steps.

A Markov chain is *ergodic* if it is *irreducible*, i.e., for all $x, y \in \Omega$, there is a t

such that $P^t(x,y) > 0$, and *aperiodic*, i.e., for all $x, y \in \Omega$, g.c.d.$\{t : P^t(x,y) > 0\} = 1$. Any finite, ergodic Markov chain converges to a unique distribution π, i.e., for all $x, y \in \Omega$, $\lim_{t\to\infty} P^t(x,y) = \pi(y)$. In fact, for any distribution π' such that $\pi'(x)P(x,y) = \pi'(y)P(y,x)$ (the *detailed balance condition*) and $\sum_{x\in\Omega} \pi'(x) = 1$, π' is the unique stationary distribution of $\mathcal{M}$ (see, e.g., [13]).

Given a desired stationary distribution π on Ω, the celebrated Metropolis-Hastings algorithm [14] defines appropriate transition probabilities. Starting at state x, pick a neighbor y in Ω uniformly with probability $1/(2\Delta)$, where Δ is the maximum number of neighbors of any state, and move to y with probability $\min(1, \pi(y)/\pi(x))$; with the remaining probability stay at x and repeat. Using detailed balance, one can verify if the state space is connected then π must be the stationary distribution. While calculating $\pi(x)/\pi(y)$ seems to require global knowledge, this ratio can often be calculated using only local information when many terms cancel out.

The motivation underlying the designs of the Markov chains we introduce is from statistical physics, where ensembles of particles reminiscent of our Amoebot model are used to study physical systems. Like a spring relaxing, systems tend to favor configurations that minimize energy. The energy function is determined by a *Hamiltonian* $H(\sigma)$; each configuration σ has weight $w(\sigma) = e^{-B\cdot H(\sigma)}/Z$, where where B is inverse temperature and $Z = \sum_\tau e^{-B\cdot H(\tau)}$ is the normalizing constant known as the *partition function*.

For algorithms running on the Amoebot model, we assign each configuration σ a Hamiltonian $H(\sigma) = -\sum_{x\in E(\sigma)} f(x)$, where $E(\sigma)$ is the set of edges in σ, i.e., the set of edges of the triangle lattice G_Δ with both endpoints occupied by particles. The function f from edges to real numbers is used to conditionally assign different weights to edges based on the problem at hand.

2.4 Simulations

To fine tune the rules and weights of the algorithms as well as to understand their behavior, extensive simulations of each algorithm were completed on a variety of different particle system configurations. These simulations were run on *Particles*, a simulator developed as part of this project, available at [15].

CHAPTER 3
PARTICLE SEPARATION

3.1 Introduction

The segregation and integration of heterogeneous sets of entities is a behavior that is observed in many places in nature. Molecules exhibit attraction and repulsion, inherent biases affect how humans form and split into social groups, and species that often peacefully share space with others choose to regroup and prioritize their survival.

We investigate this phenomenon as it applies to *self-organizing particle systems*. Here we consider *heterogeneous* particle systems — where particles have different immutable *colors* — and seek local, distributed algorithms which, when executed independently and concurrently by all particles, result in observable *separation* or *integration* of color classes regardless of the starting state.

Our work harnesses the relationships between local preferences and global phenomena to develop a stochastic, distributed, local, asynchronous algorithm that enables our particle systems to move between segregated and integrated configurations using only a simple change in parameters.

3.1.1 Systems of Heterogeneous Particles

We consider a particle system composed of n heterogeneous particles, generalizing previous work in which all particles were identical and indistinguishable [5, 6]. We model heterogeneity by assuming each particle P keeps a color $c(P) \in \{c_1, \ldots, c_k\}$ in its memory that is visible to itself and its neighbors, where $k < n$ is some small constant. For simplicity, we assume $k = 2$ in this paper, though we expect our ideas, algorithms, and proof techniques will generalize to larger k with little additional effort. The colors are

meant to represent class identity: for example, demographic diversity in human communities, the particular species a certain animal in nature belongs to, or the element of which an atom is an instance.

We say an edge between particles is *homogeneous* if both incident particles are of the same color. We also extend the notation for neighborhoods $N(\ell)$ to heterogeneous systems: for a location ℓ, let $N_i(\ell)$ denote the set of particles of color c_i adjacent to location ℓ.

We define a *swap move* under the Amoebot model that enables adjacent particles of different colors to switch places. Adding this natural swap move allows faster convergence of our algorithms in practice, but is not necessary for any results we present.

3.1.2 Results

We present a *Markov chain* $\mathcal{M}$ for particle system separation that can be directly translated into a *fully distributed, local, asynchronous compression algorithm* $\mathcal{A}$. Both $\mathcal{A}$ and $\mathcal{M}$ take as input two *bias parameters*: λ for compression (where $\lambda > 1$ makes induced lattice edges more favorable) and γ for separation (where $\gamma > 1$ makes *homogeneous* induced lattice edges more favorable) and start from an arbitrary initial configuration for the particles that is connected and has no holes.

The design of the Markov chain $\mathcal{M}$ ensures that the particles always stay connected without holes. We prove that $\mathcal{M}$ is ergodic, allowing the application of many of the standard Markov chain analysis tools.

Finally, we prove that, among configurations with small perimeter, those that are separated are exponentially more likely than those that are not in the stationary distribution of $\mathcal{M}$, which is also the stationary distribution of $\mathcal{A}$.

3.1.3 Related Work

We use a Markov chain to develop our distributed algorithm for separation, and there are several relevant related works in that area. Of particular interest is the classical Schelling

housing model [16, 17] and related models from statistical physics, such as the Ising model of ferromagnetism [18, 19]. In these models, agents (or "sites") are assigned one of two colors (or "spins"), and each agent prefers to have the same color as its neighbors. Depending on the strength of this preference, an agent may change its color or move to a new location in order to agree with more of its neighbors. These models undergo a phase transition with respect to the preference parameter: at high values the two colors are well integrated, while at low values large monochromatic regions appear. Several variants of the Schelling model have recently been shown to exhibit similarly interesting behavior [20, 21] and have received attention from the distributed computing community [22]. Our work also considers local neighborhood information and utilizes threshold-like mechanics, but with three key differences: (i) our particles cannot change their color, so the number of particles of each color is fixed; (ii) we do not assume that every position is occupied as the Schelling model does; and (iii) particles can only move to neighboring locations. Taken together, these constraints define a different and potentially more interesting set of dynamics.

3.1.4 Problem Definition

Our goal is to find a local, distributed algorithm that results in *separation* or *integration* of color classes.

Informally, we say a particle configuration with two colors is *separated* if we can identify a set R of particles such that R mostly contains particles of color c_1, its complement $\overline{R}$ mostly contains particles of color c_2, and the boundary between R and $\overline{R}$ is small. If this is the case, we say R and $\overline{R}$ are *clusters*. To further quantify this, we say a configuration is (β, δ)-clustered, for $\beta > 0$ and $\delta < 1/2$, if there are at most $\delta|R|$ particles of color c_2 in R , at most $\delta|\overline{R}|$ particles of color c_1 in $\overline{R}$, and the boundary between R and $\overline{R}$ is of size at most $\beta\sqrt{n}$, where n is the total number of particles.

In the *separation problem*, we consider an instance $(\sigma_0, \beta, \delta)$, where σ_0 is an initial configuration of colored particles and $\beta > 0$ and $0 < \delta < 1/2$ are constants. We say

a distributed algorithm solves an instance of the separation problem if, *beginning from configuration σ_0, with all but exponentially small probability it reaches and remains in a set of configurations that are (β, δ)-clustered*. When the boundary of particle configurations is fixed and small our distributed algorithm $\mathcal{A}$ provably solves the separation problem for any σ_0 with two colors, any $\beta > 4$, and any $\delta < 1/2$. Algorithm $\mathcal{A}$ also achieves integration in simulation when using different values of the input parameters, but we do not give rigorous guarantees.

3.2 A Stochastic Algorithm for Separation

In order to achieve sepearation, our separation algorithm uses two biases λ and γ to control how strongly particles prefer neighbors and same-color neighbors, respectively.

The following locally-checkable properties, first introduced in [5], ensure the particle system configuration remains connected and hole-free.

Property 1. *$|\mathbb{S}| \in \{1, 2\}$ and every particle in $N(\ell \cup \ell')$ is connected to exactly one particle in $\mathbb{S}$ by a path through $N(\ell \cup \ell')$.*

Property 2. *$|\mathbb{S}| = 0$, and both $N(\ell) \setminus \{\ell'\}$ and $N(\ell') \setminus \{\ell\}$ are nonempty and connected.*

These properties need not be verified for swap moves, which do not change the set of occupied nodes and thus cannot disconnect the system or create a hole. We can now introduce the Markov chain $\mathcal{M}$ for an instance $(\sigma_0, \beta, \delta)$ of the separation problem. For input parameters λ, γ and an initial configuration σ_0 which we assume is connected and hole-free, repeat Algorithm 1. If σ_0 has holes, $\mathcal{M}$ will eliminate them and they will not reform; we focus only on what happens after this occurs. The state space Ω of $\mathcal{M}$ contains all connected, hole-free configurations with the same number of particles of each color as σ_0.

Algorithm 1 Markov Chain $\mathcal{M}$ for Separation and Integration

Choose particle P uniformly at random from all n particles; let c_i be its color and ℓ its location.
Choose a neighboring location ℓ' and $q \in (0, 1)$ uniformly at random.
if ℓ' is unoccupied **then**
    P expands to occupy both ℓ and ℓ'.
    if (i) ℓ and ℓ' satisfy Property 1 or 2 and (ii) $q < \lambda^{|N(\ell')|-|N(\ell)|} \cdot \gamma^{|N_i(\ell')|-|N_i(\ell)|}$ **then**
        P contracts to ℓ'.
    else P contracts back to ℓ.
else if ℓ' is occupied by particle Q of color c_j **then**
    P calculates $|N_i(\ell)|$ and $|N_j(\ell) \setminus \{Q\}|$ and sends these values to Q.
    Q calculates $|N_i(\ell') \setminus \{P\}|$ and $|N_j(\ell')|$.
    if $q < \gamma^{|N_i(\ell')\setminus\{P\}|-|N_i(\ell)|+|N_j(\ell)\setminus\{Q\}|-|N_j(\ell')|}$ **then** Q swaps with P.

However, we note that the Markov chain $\mathcal{M}$ still requires global knowledge to execute and is centralized. To translate Markov chain $\mathcal{M}$ into a fully local, distributed, asynchronous algorithm $\mathcal{A}$ that is run by each particle concurrently, we decompose the steps of $\mathcal{M}$ into individual particle activations, in which a single particle performs some computation and at most one movement [23]. Details of this decomposition can be found in [24], and since the decomposition proves $\mathcal{M}$ and $\mathcal{A}$ to have the same stationary distribution, we continue our analysis with $\mathcal{M}$.

3.3 Properties of the Algorithm

We now examine the behavior of $\mathcal{M}$, and — by extension — the behavior of $\mathcal{A}$.

Lemma 3.3.1. *If σ_0 is a connected, hole-free configuration, then the current configuration at every iteration of $\mathcal{M}$ is also connected and hole-free.*

Proof. Moves of a particle into an unoccupied location are limited by Properties 1 and 2

exactly as in [5], and their results show such moves cannot disconnect the system or create a hole. Swap moves also cannot disconnect the system or form a hole since they leave the set of occupied vertices unchanged. These are the only two types of moves allowed by $\mathcal{M}$. □

Lemma 3.3.2. *Markov chain $\mathcal{M}$ is ergodic.*

Proof. First, $\mathcal{M}$ is aperiodic: there is always a positive probability of a swap between similarly colored particles, which does not change the state of $\mathcal{M}$. To see that the moves of $\mathcal{M}$ suffice to move between any states of Ω, we note the set of non-swap moves of $\mathcal{M}$ is the same as in [5]. Their results show that for homogeneous particle systems these moves suffice to move between any two connected hole-free configurations. If two configurations are the same when colors are disregarded, swap moves can then be used to transform one into the other without changing the overall shape. As a result, allowed moves can transform any configuration σ into any other configuration τ in Ω, so $\mathcal{M}$ is irreducible and thus ergodic. □

As $\mathcal{M}$ is finite and ergodic, it converges to a unique stationary distribution π which we can find using detailed balance.

Lemma 3.3.3. *The stationary distribution of $\mathcal{M}$ is given by $\pi(\sigma) = \lambda^{e(\sigma)} \cdot \gamma^{a(\sigma)}/Z$, where $Z = \sum_{\sigma} \lambda^{e(\sigma)} \cdot \gamma^{a(\sigma)}$.*

Proof. Because $\mathcal{M}$ is ergodic with a finite state space, it converges to a unique stationary distribution. We verify this stationary distribution is π via detailed balance, that is, by showing for all $\sigma, \tau \in \Omega$ that $\pi(\sigma)P(\sigma,\tau) = \pi(\tau)P(\tau,\sigma)$, where P is the transition matrix of $\mathcal{M}$. It suffices to consider pairs σ, τ where $P(\sigma,\tau) > 0$; because we have carefully defined $\mathcal{M}$ to be reversible, especially Properties 1 and 2, this happens if and only if $P(\tau,\sigma) > 0$. There are two cases to consider, one for each type of move allowed by $\mathcal{M}$.

If σ and τ differ by a move of particle P of color c_i from location ℓ in σ to location ℓ'

in τ, then Steps 1–6 of Algorithm 1 imply

$$P(\sigma,\tau) = \frac{1}{6n}\min\{1, \lambda^{|N(\ell')|-|N(\ell)|}\cdot\gamma^{|N_i(\ell')|-|N_i(\ell)|}\}.$$

$$P(\tau,\sigma) = \frac{1}{6n}\min\{1, \lambda^{|N(\ell)|-|N(\ell')|}\cdot\gamma^{|N_i(\ell)|-|N_i(\ell')|}\}.$$

Without loss of generality, we suppose $\lambda^{|N(\ell')|-|N(\ell)|}\cdot\gamma^{|N_i(\ell')|-|N_i(\ell)|} < 1$, so the minimum in $P(\sigma,\tau)$ is this value and the minimun in $P(\tau,\sigma)$ is 1. Noting that

$$e(\sigma) - |N(\ell)| + |N(\ell')| = e(\tau) \quad \text{and} \quad a(\sigma) - |N_i(\ell)| + |N_i(\ell')| = a(\tau),$$

it follows that

$$\pi(\sigma)P(\sigma,\tau) = \frac{\lambda^{e(\sigma)}\gamma^{a(\sigma)}}{Z}\frac{1}{6n}\lambda^{|N(\ell')|-|N(\ell)|}\cdot\gamma^{|N_i(\ell')|-|N_i(\ell)|} = \frac{\lambda^{e(\tau)}\gamma^{a(\tau)}}{Z}\frac{1}{6n} = \pi(\tau)P(\tau,\sigma).$$

Now, suppose σ and τ differ by a swap move where σ has P of color c_i at location ℓ and Q of color c_j at neighboring location ℓ' and τ has P and Q in opposite positions. If c_i = c_j, then $\sigma = \tau$ and we are done. When $c_i \neq c_j$, then Steps 1, 2, and 8–11 of Algorithm 1 imply that, because this swap move could be initiated by P or by Q,

$$\begin{aligned} P(\sigma,\tau) &= \frac{1}{6n}\min\{1, \gamma^{|N_i(\ell')\setminus\{P\}|-|N_i(\ell)|+|N_j(\ell)\setminus\{Q\}|-|N_j(\ell')|}\} \\ &\quad + \frac{1}{6n}\min\{1, \gamma^{|N_j(\ell)\setminus\{Q\}|-|N_j(\ell')|+|N_i(\ell')\setminus\{P\}|-|N_i(\ell)|}\} \\ &= \frac{1}{3n}\min\{1, \gamma^{|N_i(\ell')\setminus\{P\}|-|N_i(\ell)|+|N_j(\ell)\setminus\{Q\}|-|N_j(\ell')|}\} \end{aligned}$$

For τ, which has P in position ℓ' and Q in position ℓ, we can similarly define

$$P(\tau,\sigma) = \frac{1}{3n}\min\{1, \gamma^{|N_i(\ell)\setminus\{P\}|-|N_i(\ell')|+|N_j(\ell')\setminus\{Q\}|-|N_j(\ell)|}\}$$

Due to the differing positions of P and Q in σ and τ, the exponential expressions in each

of the minimums above are inverses of each other. We note that $e(\sigma) = e(\tau)$, because the total number of edges in the particle configuration remains unchanged by a swap move. Furthermore,

$$a(\sigma) - |N_i(\ell)| - |N_j(\ell')| + |N_i(\ell') \setminus \{P\}| + |N_j(\ell) \setminus \{Q\}| = a(\tau).$$

Without loss of generality, we assume $\gamma^{|N_i(\ell')\setminus\{P\}|-|N_i(\ell)|+|N_j(\ell)\setminus\{Q\}|-|N_j(\ell')|} < 1$. This implies the minimum in $P(\tau, \sigma)$ is 1. We conclude

$$\begin{aligned}\pi(\sigma)P(\sigma,\tau) &= \frac{\lambda^{e(\sigma)}\gamma^{a(\sigma)}}{Z}\frac{1}{6n}\gamma^{|N_i(\ell')\setminus\{P\}|-|N_i(\ell)|+|N_j(\ell)\setminus\{Q\}|-|N_j(\ell')|} \\ &= \frac{\lambda^{e(\tau)}\gamma^{a(\tau)}}{Z}\frac{1}{6n} = \pi(\tau)P(\tau,\sigma).\end{aligned}$$

Detailed balance has been verified for all $\sigma, \tau \in \Omega$, so we conclude π is the stationary distribution of $\mathcal{M}$. □

Moreover, we prove that, among configurations with small perimeter, those that are separated are exponentially more likely than those that are not in the stationary distribution of $\mathcal{M}$, which is also the stationary distribution of $\mathcal{A}$. The proof of this claim, which is formalized in the theorem below, is available in [24].

Theorem 3.3.4. *For any $\alpha > 1$, $\beta > 4\alpha$, and $\delta < 1/2$, there exists γ^* and n_0 (which depend on α, β, and δ) such that for all $\gamma > \gamma^*$ and $n > n_0$, for any α-compressed boundary contour $\mathcal{P}$, the probability that a particle configuration drawn at random from $\pi_{\mathcal{P}}$ is not (β, δ)-clustered is at most $\xi^{\sqrt{n}}$ for some constant $\xi < 1$ (ξ depends on α, β, and δ).*

3.4 Simulations

To complement our rigorous analysis of the algorithm, we also provide simulations of its behavior on a system of 100 particles with 50 particles each of two colors. On the

simulations, $\mathcal{M}$ exhibits the expected separation behavior for large λ and γ, as well as integration behaviors for other parameter values. Fig. 3.1 shows the progression of $\mathcal{M}$ over time with biases $\lambda = 4$, $\gamma = 4$, a regime in which we expect the entire system to compress and individual color classes to segregate. Fig. 3.2 compares the resulting configurations after running $\mathcal{M}$ from the same initial configuration for the same number of iterations, varying only the values of λ and γ. We observe four distinct phases: expanded-integrated, expanded-separated, compressed-integrated, and compressed-separated (Fig. 3.2).

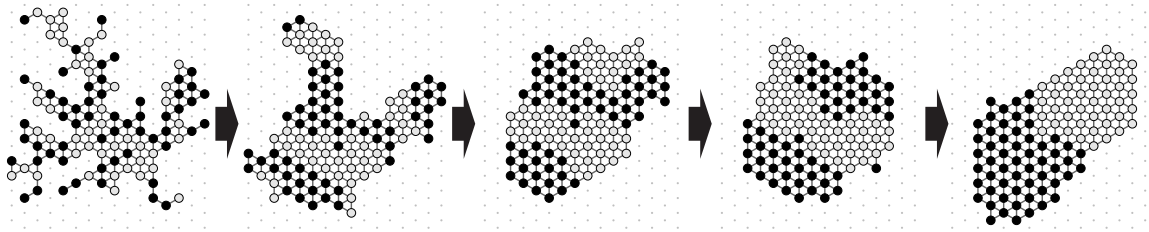

Figure 3.1: A 2-color heterogeneous particle system starting in an arbitrary state after — from left to right — 0; 50,000; 1,050,000; 17,050,000; and 68,250,000 iterations of $\mathcal{M}$ with $\lambda = 4$ and $\gamma = 4$.

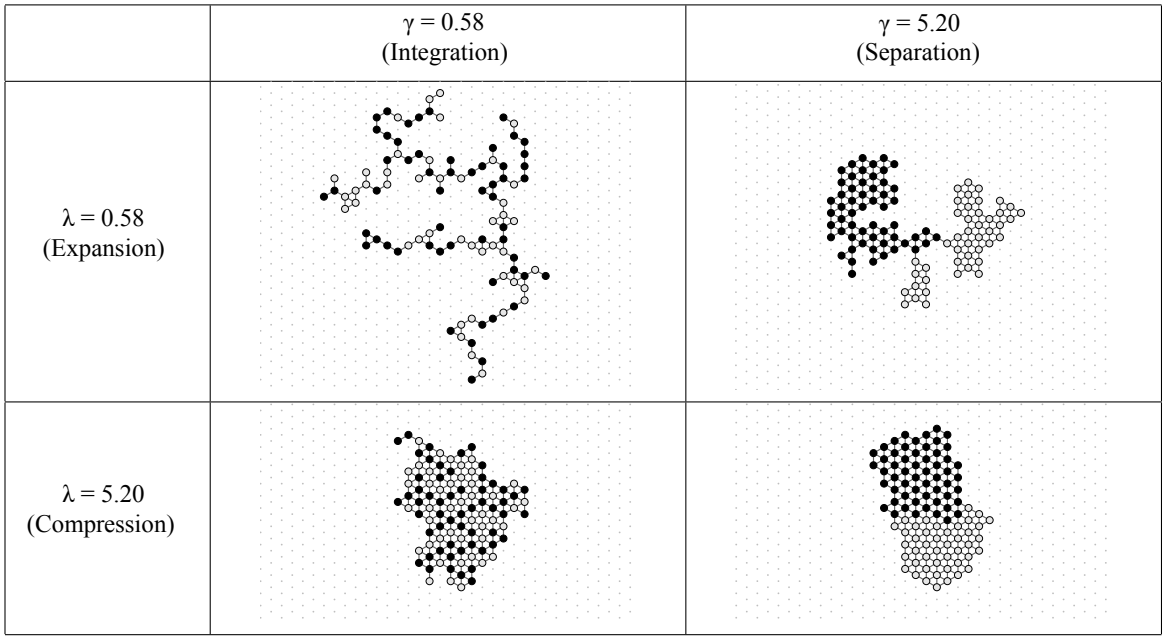

Figure 3.2: A 2-color heterogeneous particle system starting in the leftmost configuration of Fig. 3.1 after 50 million iterations of $\mathcal{M}$ for various values of the parameters λ and γ.

CHAPTER 4
PARTICLE ALIGNMENT AND FLOCKING

4.1 Introduction

A collective behavior seen often in nature is *flocking*, where organisms of the same species orient themselves in a given direction and move in a coordinated manner. This behavior is seen in the flocking of birds, schooling of fish, and herd behavior of land animals [25], having advantages in terms of helping defend the group of organisms from external threats by keeping them together as well as maintaining a high level of efficiency during the actual movement by benefiting from aerodynamic effects.

While the flocking problem has been investigated in the past decades with great success in producing algorithms that seem to replicate the behavior with high realism, claims about the behavior of most flocking algorithms are based on experimental evidence [25]. We aim to investigate the flocking behavior as it applies to *directed* self-organizing particle systems, producing a Markov chain algorithm that accurately replicates this behavior and that can be analyzed to provide rigorous guarantees of behavior.

We model a directed self-organizing particle system by adding a *direction* flag that we use to indicate the direction a particle is *facing*. On the triangular lattice, the direction flag is chosen from the set of six directions that the particle has a neighbor in. We use the notation $d(P)$ to refer to the direction a particle is facing.

4.2 Preliminary Results

Preliminary work on the problem has focused on an algorithm consisting of two moves: *spins* and *translations*. When activated, a particle chooses at random to do either move.

If a particle chooses a spin move, it uses bias parameter λ to determine the weight

of each direction d it can face, by summing the similarity measure $sim(a, b)$ (currently defined as the cosine function normalized to the range $[0, 1]$) of d with the direction $d(P)$ and calculating $\lambda^{\sum_{P' \in N(P)} sim(d,d(P'))}$. Then it uses this weight to pick a direction at random, and updates its direction to equal this newly sampled direction.

If a particle chooses a translation move, it uses a bias parameter γ to pick a movement direction at random, where each direction d is weighted based on its similarity to the particle's facing direction: $\gamma^{sim(d,d(P))}$. Then the rules of the compression algorithm in [5] are applied to produce hole-free, reversible, ergodic compression.

We display the results of preliminary simulations on the algorithm:

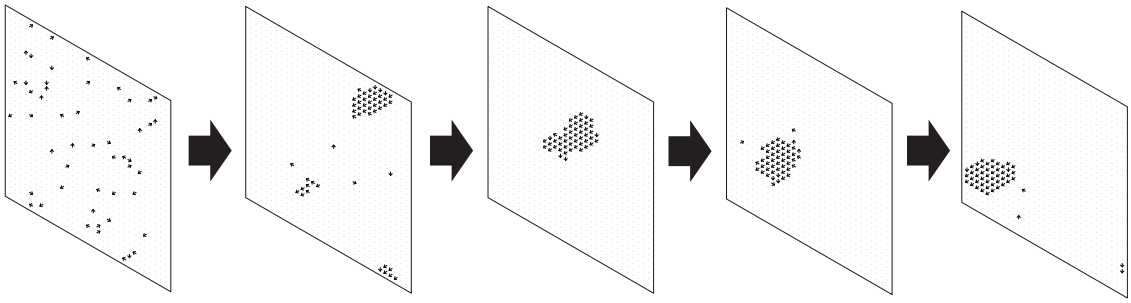

Figure 4.1: A self-organizing particle system of 50 directed particles, starting at a random configuration, after — from left to right — 0; 50,000; 300,000; 700,000; and 1,000,000 iterations of the algorithm with $\lambda = 20$ and $\gamma = 5$. After the clusters merge, all particles end up facing close to the southwest, producing drift in this direction.

4.3 Future Work

Future work on the alignment problem will focus on reaching a concrete definition of the algorithm and analyzing the properties of the stationary distribution of the algorithm in order to reach rigorous guarantees of its flocking behavior. Different models, including ones with continuous facing directions and placement on $\mathbb{R}^2$ rather than the triangular lattice, will be investigated, to provide a flexible flocking model that can be applied in production artificial intelligence contexts.

REFERENCES

[1] J. J. Daymude, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann, "Improved leader election for self-organizing programmable matter," in *Algorithms for Sensor Systems*, A. Fernández Anta, T. Jurdzinski, M. A. Mosteiro, and Y. Zhang, Eds., Cham: Springer International Publishing, 2017, pp. 127–140, ISBN: 978-3-319-72751-6.

[2] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann, "An algorithmic framework for shape formation problems in self-organizing particle systems," in *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication*, New York, NY, USA: ACM, 2015, 21:1–21:2, ISBN: 978-1-4503-3674-1.

[3] A. Porter and A. W. Richa, "Collaborative computation in self-organizing particle systems," To be submitted, available online at `https://arxiv.org/abs/1710.07866`, 2018.

[4] Z. Derakhshandeh, R. Gmyr, A. W. Richa, C. Scheideler, and T. Strothmann, "Universal coating for programmable matter," *Theoretical Computer Science*, vol. 671, pp. 56–68, 2017.

[5] S. Cannon, J. J. Daymude, D. Randall, and A. W. Richa, "A Markov chain algorithm for compression in self-organizing particle systems," in *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC '16)*, 2016, pp. 279–288.

[6] M. Andrés Arroyo, S. Cannon, J. J. Daymude, D. Randall, and A. W. Richa, "A stochastic approach to shortcut bridging in programmable matter," in *DNA Computing and Molecular Programming — 23rd International Conference (DNA23)*, A full and updated version is available at `https://arxiv.org/abs/1709.02425`, 2017, pp. 122–138.

[7] R. J. Baxter, I. G. Enting, and S. K. Tsang, "Hard-square lattice gas," *Journal of Statistical Physics*, vol. 22, no. 4, pp. 465–489, Apr. 1980.

[8] A. Blanca, D. Galvin, D. Randall, and P. Tetali, "Phase coexistence and slow mixing for the hard-core model on $\mathbb{Z}^2$," in *17th International Workshop on Randomization and Computation (RANDOM '13)*, 2013, pp. 379–394.

[9] R. Restrepo, J. Shin, P. Tetali, E. Vigoda, and L. Yang, "Improving mixing conditions on the grid for counting and sampling independent sets," *Probability Theory and Related Fields*, vol. 156, pp. 75–99, 2013.

[10] Z. Derakhshandeh, S. Dolev, R. Gmyr, A. Richa, C. Scheideler, and T. Strothmann, "Brief announcement: Amoebot - a new model for programmable matter," in *26th ACM Symp. on Parallelism in Alg. and Arch., SPAA '14*, 2014, pp. 220–222.

[11] J. Daymude, Z. Derakhshandeh, R. Gmyr, A. Porter, A. Richa, C. Scheideler, and T. Strothmann, "On the runtime of universal coating for programmable matter," *Natural Computing*, pp. 1–16, Nov. 2017.

[12] N. Lynch, *Distributed Algorithms*. Morgan Kauffman, 1996.

[13] W. Feller, *An Introduction to Probability Theory and Its Applications*. Wiley, 1968, vol. 1.

[14] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[15] C. Gokmen, *Particles*, `https://github.com/skyman/particles`, 2017.

[16] T. C. Schelling, "Models of segregation," *The American Economic Review*, vol. 59, no. 2, pp. 488–493, 1969.

[17] ——, "Dynamic models of segregation," *The Journal of Mathematical Sociology*, vol. 1, no. 2, pp. 143–186, 1971.

[18] D. Randall, "Rapidly mixing Markov chains with applications in computer science and physics," Computing in Science and Engineering, vol. 8, pp. 30–41, 2006.

[19] D. Vinković and A. Kirman, "A physical analogue of the schelling model," *Proceedings of the National Academy of Sciences*, vol. 103, no. 51, pp. 19 261–19 265, 2006.

[20] P. Bhakta, S. Miracle, and D. Randall, "Clustering and mixing times for segregation models on $\mathbb{Z}^2$," in *25th ACM-SIAM Symposium on Discrete Algorithms*, 2014, pp. 327–340.

[21] N. Immorlica, R. Kleinberg, B. Lucier, and M. Zadomighaddam, "Exponential segregation in a two-dimensional schelling model with tolerant individuals," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '17)*, 2017, pp. 984–993.

[22] H. Omidvar and M. Franceschetti, "Self-organized segregation on the grid," in *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC '17)*, 2017, pp. 401–410.

[23] J. J. Daymude, A. W. Richa, and C. Scheideler, "The amoebot model," Available online at `https://sops.engineering.asu.edu/sops/amoebot`, 2017.

[24] S. Cannon, J. J. Daymude, C. Gokmen, D. Randall, and A. W. Richa, "A local stochastic algorithm for separation in heterogeneous self-organizing particle systems," Available on arXiv, 2018.

[25] B. Chazelle, "Natural algorithms," in *Proceedings of the 2009 ACM-SIAM Symposium on Discrete Algorithms (SODA '09)*, 2009, pp. 422–431.